Text Classification with RNN and LSTM models using Python

Project Overview

Business Overview

Text Classification is one of the essential applications of Natural Language Processing. Neural network-based methods have obtained significant progress on various natural language processing tasks. As deep learning is emerging, training complex data has become faster and easier with networks like RNNs and LSTM.

In the previous projects, we have witnessed how to use the Naïve Bayes algorithm for text classification, and we have also implemented the skip-gram model for word embeddings. This project will see how to implement the Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) models for text classification.

Aim

To perform text classification on the dataset using RNN and LSTM models.

Data Description

The dataset contains more than two million customer complaints about consumer financial products. Amongst the various available columns, we have a column that contains the actual text of the complaint and one column containing the product for which the customer is raising the complaint. We have also used pre-trained word vectors - glove.6B. The link to download this data is as follows:
https://nlp.stanford.edu/projects/glove/

Tech Stack

➔ Language: Python
➔ Libraries: pandas, torch, nltk, numpy, pickle, re, tqdm, sklearn

Prerequisite

1. The torch framework
2. Multiclass Text Classification using Naive Bayes in Python
3. Skip Gram Model Python Implementation for Word Embeddings

Approach

1. Installing the necessary packages through pip command
2. Importing the required libraries
3. Defining configuration file paths
4. Process glove embeddings
    - Read the text file
    - Convert embeddings to float array
    - Add embeddings for padding and unknown items
    - Save embeddings and vocabulary
5. Process Text data
    - Read the CSV file and drop the null values
    - Replace duplicate labels
    - Encode the label column and save the encoder and encoded labels
6. Data Preprocessing
    - Conversion to lower case
    - Punctuation removal
    - Digits removal
    - Additional spaces removal
    - Tokenization
7. Building Data loader
8. Model Building
    - RNN architecture
    - LSTM architecture
    - Train function
    - Test function
9. Model training
    - RNN model
    - LSTM model
10. Prediction on Test data

Modular Code Overview

```
Input
   |_complaints.csv
   |_glove.6B.50d.txt


Output
   |_embeddings.pkl
   |_label_encoder.pkl
   |_labels.pkl
   |_model_lstm.pkl
   |_model_rnn.pkl
   |_vocabulary.pkl
   |_tokens.pkl

Source
   |_data.py
   |_model.py
   |_utils.py


|_config.py
|_Engine.py
|_predict.py
|_processing.py
|_README.MD
|_requirements.txt
|_RNN.ipynb
```

Once you unzip the modular_code.zip file, you can find the following folders within it.

1. Input
2. Output
3. Source

1. The input folder contains the data that we have for analysis. In our case, it
    - complaints.csv
    - glove.6B.50d.txt (download it from the link provided)

2. The source folder contains all the modularized code for all the above steps in a modularized manner. It includes the following.
    - model.py
    - data.py
    - utils.py

These all-python files contain helpful functions which are being used in the Engine.py file.

3. The output folder contains the following;
    - embeddings.pkl
    - label_encoder.pkl
    - labels.pkl
    - model_lstm.pkl
    - model_rnn.pkl
    - vocabulary.pkl
    - tokens.pkl

    These are required for our model training and can be quickly loaded and used for future use, and the user need not have to train all the models from the beginning. (model_lstm.pkl and model_rnn.pkl are our saved models after training)

4. The config.py file contains all the configurations required for this project.

5. The Engine.py file is the main file that needs to be called to run the entire code in one go. It trains the model and saves it in the output folder.
    Note: Please check the README.md file for more information.

6. The RNN.ipynb is the original notebook we saw in the videos.

7. The processing.py file is used to process the data.

8. The predict.py file is used to make predictions on the data.

9. The README.md file contains all the information on how to run particular files and more instructions to follow.

10. The requirements.txt file has all the required libraries with respective versions. Kindly install the file by using the command **pip install -r requirements.txt**

Project Takeaways

1. What are pre-trained word vectors?
2. What is a Recurrent Neural Network (RNN)?
3. Understanding the working of the RNN network
4. What is the vanishing gradient problem?
5. What is Long Short-Term Memory (LSTM)?
6. Understanding the working of LSTM networks.
7. Steps to process glove embeddings
8. Data preparation for the models
9. How to remove spaces, digits?
10. Removing the punctuations
11. How to create a data loader for the RNN and LSTM models?
12. Building RNN model
13. Building LSTM model
14. Training RNN and LSTM model using CUDA or CPU
15. How to make predictions on the new text data?